Malware Detection in Android Devices

Andrew Montgomery
Computer Science and Software Engineering
University of Washington

***Abstract* – Android devices are the most popular mobile device on the market. The prevalence of Android devices has drawn the attention of malicious actors and are now a primary target for malware developers. Malware developers have developed numerous techniques to penetrate Android devices and avoid detection. Malware detection methods have also evolved to maintain pace with malware developers.**
This paper contributes a comprehensive review of Android malware and associated detection methods. To understand the threat that malware poses to Android users, we believe it is important to understand the most significant threats currently available and the penetration methods that are being used to infect devices. With that information, a review of existing and emerging detection methods can be analyzed to aid in the determination of the best potential salutation for protecting Android devices from malware.

I. Introduction

Smartphones today are mobile personal computers, performing many of the same functions as desktop computers: internet browsing, games, social media, online banking, and more. Smartphones also implement many features unique to phones, such as SMS messaging, location data, and different connectivity options (Wi-Fi, GPS, Bluetooth, GSM). The functionality as well as the convenience provided by smartphones has led smartphones to be a must have technology, and thus makes smartphones an ideal target for malware developers.

During I/O 2019, Google revealed that it now powers 2.5 billion devices per month, and between May 2017 and May 2019, Google added 500 million additional devices, or roughly 20 million devices per month [1]. According to the International Data Corporation, Android now holds 86 percent of the worldwide smartphone market [2]. Figure 1 shows the growth of the Android market share from 2009 through 2017. Android continues to grow each year and dominate the mobile market.

Android devices offer the same functionalities as other mobile device platforms, but at a reduced cost, making Android accessible to more people. The increased accessibility has led to Android devices becoming a prime target for malicious attackers targeting mobile devices.

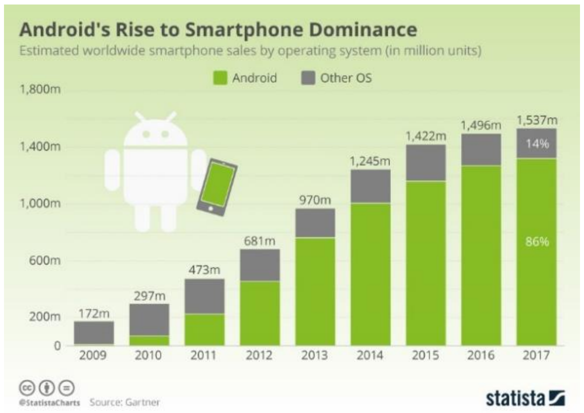


Figure 1: Android Market Share [3]

Android devices are more susceptible to vulnerabilities because the Android OS is open source. Android manufacturers can make changes to the Android OS to fit their own needs. According to the fourth annual Android fragmentation report by OpenSignal, there was a total of 24,093 different Android devices made by 1,294 companies in 2015 [4]. This fractured manufacturing ecosystem creates a major problem for developers because there is no single application development standard to follow. Developers must understand and try to make applications compatible with a wide range of hardware and software configurations.

According to Statista, as of March 2020, the number of available apps in the Google Play Store was about 2.8 million applications [5]. Sensor Tower released a report based on Google Play Store statistics for 2019 – 2020, where they estimated the number of application downloads per day to be 250 million [6]. Third-party developers create most of the applications available in the Google Play Store.

Google Play does not manually verify applications published by developers, instead relying on the Bouncer system to identify malicious applications [7]. However, the Bouncer system is ineffective and easily bypassed by attackers.

Android devices allow the download and use of applications from third-party application stores, where verification of applications and detection of malicious behavior may be inexistent or less effective than on the official Google Play Store. Malware developers will often repackaged legitimate popular applications from the Google Play Store, by injecting them with malicious code, and then distributing them on third-party application stores [8]. Often, smartphone users believe that these applications are inherently safe, and do not properly verify the source of the downloaded application.

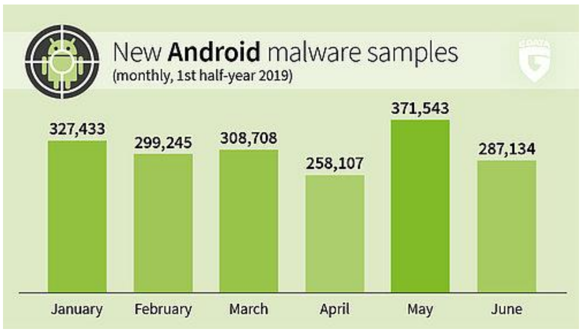


Figure 2: New Android Malware Samples [10]

Mobile malwares include Trojans, adware, spyware, and other malicious programs that steal personal and financial information. Malware developers avoid detection in the application stores using code obfuscation, repackaging, encryption, dynamic execution, and stealth techniques. For example, CamScanner was an application with more than 100 million downloads in the Google Play Store that contained malware. Kaspersky products detected this module to be a Trojan dropper, meaning the module extracts and runs another malicious module from an encrypted file included in the application's resources. This "dropped" malware, is a Trojan downloader that downloads additional malicious modules that can then show intrusive adds or sign users up for paid subscriptions to external services [9]. This malicious activity was not originally in the application on release, and once discovered, Google Play removed the app. However, millions of users had already downloaded the malicious app to their device.

The previous example is only one way in which malware developers are attacking Android devices. These other attacks include various trojans, ransomware, mobile botnets, cryptojacking, among others. The amount of sensitive data that is present on Android devices makes malware detection and protecting user's data a major concern. The threat that malware poses to Android devices has led to antimalware companies developing various techniques for detecting malicious activity. Two main categories of malware detection in Android devices are a static approach and dynamic approach, but current research has focused on the utilization of machine learning and blockchain technology to enhance the effects of these two approaches.

In this paper, we aim to do a comprehensive review of malware in Android devices. The paper will discuss the different types of mobile malwares, the common penetration methods of mobile malware, evasion techniques used by malware developers, and an extensive review of existing and emerging malware detection methods. In addition, we provide a comparative analysis of the existing and emerging malware detection methods to aid in determining the most effective solutions. The paper concludes by discussing the future of malware detection research. The organization of this survey paper is as follows:

- Section II discusses the most current malware threats
- Section III explores the various malware penetration methods and evasion techniques
- Section IV reviews the existing and emerging malware detection methods
- Section V provides a comparative analysis and discussion of the current and emerging detection methods
- Section VI concludes the paper and details the future directions of Android malware detection

II. Android Malware

According to a report released by G Data Security [10], In the first half of 2019, researchers found 1.85 million new malicious apps in Android devices. The large number of new malicious apps demonstrates that mobile malware poses a significant threat to Android users. Antimalware applications have detected a wide range of mobile malwares. Figure 2 shows the number of new Android malware samples discovered in each month of the first half of 2019.

A. Trojans

A trojan is a type of malicious application that appears to be legitimate. When a user downloads and executes a malicious application, trojan's use a variety of mechanisms to disrupt and steal the user's private data. Discovered roughly ten years ago, *SpyEye* [11] banking trojan was the first banking trojan for Android devices. SpyEye works in conjunction with the SpyEye malware for windows, and enabled attackers to bypass two-factor authentication to gain access to mobile banking information. *FakeInst* [12] is a trojan that poses as a premium application, but after downloading, installing, and running the application prompts the user to send a text message to access paid features of the application. After gaining access to the users' phone number, the app will then send premium SMS messages. Current banking trojans pose an even more serious threat. *BankBot* [13], not only can steal users' banking credentials, but can also intercept incoming SMS messages, steal phone book information, gather data from the infected device, and uninstall some antivirus programs.

B. Mobile Ransomware

Ransomware is a malicious application that denies the full use or access to information on the users' device. Attackers seek financial payment, generally in some form of cryptocurrency, to restore access to the device. The primary difference between mobile ransomware compared to traditional ransomware is that the typical encryption process often fails due to the prevalence of cloud storage, as well as the limitations of encryption created by mobile CPUs and battery life [14]. The most recent mobile ransomwares lock access to the entire device until the user sends payment. *Koler* [15] ransomware posed as an adult orientated app, that when downloaded would gain admin access to the infected device. The attacker would then display a message posing as the FBI and demand payment from the user.

C. Cryptojacking

With the explosion of cryptocurrencies, such as bitcoin, in 2018, mobile malware developers saw potential for financial gain by using infected mobile devices to mine cryptocurrencies. Cryptomining uses computer resources, such as CPU cycles, to "mine" the currency. Cryptomining malwares give attackers access to these resources available in Android devices. Malicious attackers often package cryptomining malware with trojans or ransomware to ensure financial gain if the attacker cannot extract the user's private info or other form of payment. *UFO Miner* [16] is an example of one such application. The application has no user interface and does not show up in the application list of an Android device.

D. Adware

Adware hijacks Android devices to perform fake clicks on ads to generate income for the attacker. The World Federation of Advertisers [17] estimated that ad fraud could generate up to $50 billion by 2025. Adware works similarly to cryptomining malware, operating silently in the background. This malware does not necessarily pose a significant threat to users' private information but can have negative effects on Android devices' performance and use up mobile data. Malware developers pair adware with legitimate apps because adware can go undetected in Android devices if the application remains downloaded to the phone. Some popular known malicious Android adware applications are: *Snow Heavy Excavator Simulator* (10 million installs) and *Deleted Photo Recovery* (1 million installs) [18].

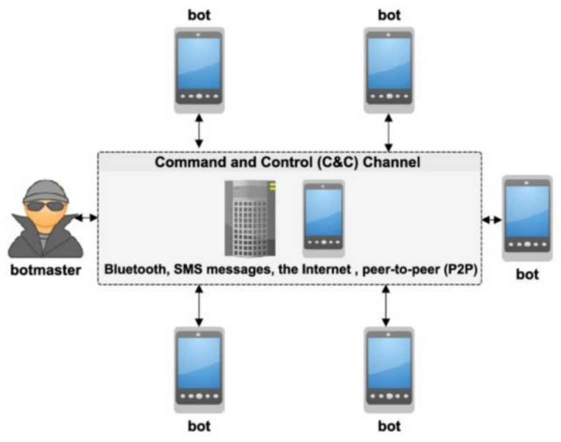

bot
bot
Command and Control (C&C) Channel
botmaster
Bluetooth, SMS messages, the Internet , peer-to-peer (P2P)
bot
bot
bot

Figure 3: Example of a Mobile Botnet [20]

E. Mobile Botnets

Mobile botnets specifically target smartphones and other mobile devices. They attempt to gain complete access to the device, and once achieved can propagate by sending copies of themselves by sending text-messages or email messages to other devices. Attackers leverage mobile botnets to perform DDoS attacks. *DroidDream* [19] appeared in the spring of 2011 and worked by gaining root access to the Android device to access unique identification information for the phone. An infected device could also download additional malicious applications without the user's knowledge. Figure 3 shows the framework of a mobile botnet.

III. Malware Penetration Methods

This section summarizes the penetration techniques used by Android malware.

A. Repackaging Attacks

A repackaging attack is when a malicious user obtains a copy of a popular application from the Google Play Store, and then injects malicious functionality into the application. The attacker then redistributes the new version of the application to unsuspecting users who believe the application is legitimate. By repackaging an application, the attacker can avoid detection by antimalware applications. Figure 4 demonstrates the repackaging process.

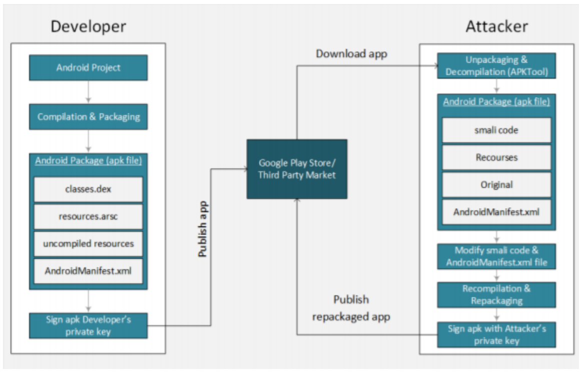

Developer
Attacker
Android Project
Compilation & Packaging
Android Package (apk file)
classes.dex
resources.arsc
uncompiled resources
AndroidManifest.xml
Sign apk Developer's private key
Publish app
Google Play Store/ Third Party Market
Download app
Unpackaging & Decompilation (APKTool)
Android Package (apk file)
smali code
Recourses
Original
AndroidManifest.xml
Modify smali code & AndroidManifest.xml file
Recompilation & Repackaging
Sign apk with Attacker's private key
Publish repackaged app

Figure 4: Application Repackaging Process [21]

Repackaging attacks are a common penetration technique used for the distribution of trojan's on Android devices. Nearly 80 percent of the top 50 free applications have repackaged versions available on third-party stores [22]. The repackaging process has four main steps, detailed below:

- **Unpacking:** Tools such as apktools extract legitimate application's APK files
- **Disassembly:** The attacker modifies the application by disassembling Dalvik bytecode in the APK files into Smali code.
- **Code injection:** The attacker injects malicious code into the Smali code of the application by either modifying the existing Smali code or adding a new component to the application.
- **Repacking:** The final step is to rebuild the apk file using *apktool* and then sign the new apk file with *Jarsigner*.

Repackaging is a popular penetration method because it is often difficult to detect due to having a similar functionality as the original application. An example of popular repacked trojan is *DroidKungFu* [23][24].

B. Drive-by-Download

Drive-by-download refers to the unintentional download of malicious code to an Android device. This attack takes advantage of security flaws in the device and does not rely on the user to do anything to execute the malicious code. Cybercriminals use these attacks to steal and collect personal data, inject trojans, or to introduce other types of malware.

In a drive-by-download attack, malicious attackers compromise a website by embedding or injecting malicious objects inside the webpage. These malicious objects range from malicious JavaScript code, redirects, maladvertisements, and among other methods. *Youku Tudou* [25] was a fake social media site. The fake website would prompt the user to download and execute the *xRAT* trojan under the guise of an Adobe Flash Player update.

C. Dynamic Payload

Malwares can penetrate Android devices by embedding encrypted malicious content in APK resources. The malicious content is decrypted and executed on the device after the user installs the application. Dynamic payloads can also be downloaded by users from remote servers. *DroidKungFu* [23][24] has an embedded app component that enables the malware to still be functional even after the removal of the repackaged from the device.

D. Stealth Malware Techniques

Nearly all current malware in the wild employs one or more stealth techniques to avoid detection by antimalware applications. Utilizing a combination of stealth techniques enables mobile malware to go undetected due to the limited resources, such as CPU processing and battery power, available to antimalware applications. Listed below are some of the stealth techniques used to disguise malware and avoid detection:

- **Code Obfuscation:** The act of deliberately obscuring source code. This technique makes it significantly more difficult for antimalware applications to detect malicious signatures
- **Anti-Emulation:** Malware behaves differently in an emulated environment to avoid detection
- **Dynamic Loading:** Malware programmed to load only when a certain trigger has occurred.
- **Adversarial Learning:** Malware developers are developing their own machine learning models to improve existing malware, quickly develop new variants, and avoid detection

IV. Malware Detection Methods

Traditionally, there are two primary approaches for malware detection: A *static approach* and a *dynamic approach*. Current research has moved in the direction of utilizing hybrid analysis and cloud-based detection as well as utilizing machine learning and blockchain technology to enhance current detection methods.

A. Static Approach

The static-based approach aims to detect malware without the execution of the program by analyzing the source code. Static analysis techniques range from signature generation to permission analysis. Antimalware systems generate application signatures by analyzing the source code of an application, and then comparing the signatures to known signatures in a database. If the application signature matches a known malware signature, then the user can be alerted to remove the application from their device. Antimalware systems analyze permission to detect the difference between the requested permissions of an application and the required permissions to run the application. Most commercial antimalware applications rely heavily on static based malware detection. Listed below are the advantages and disadvantages of static analysis-based detection:

Advantages:

- Comprehensive databases of existing malwares are available to researchers
- Static analysis methods are fast, simple to run, and widely available
- Provides strong protection from known malware threats

Disadvantages:

- Cannot detect unknown malware types
- Discovery of new malware requires an immediate update to existing databases
- High potential for false positives due to similarities in malicious and benign applications

AndroSimilar [26] is an Android malware detection system that generates signatures by extracting statistically improbable features. The proposed system is effective against code obfuscation and repackaging attacks. The system was designed by researchers as an Android device antivirus solution.

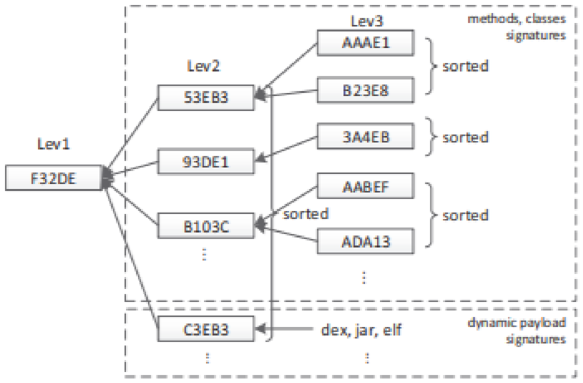


Figure 5: Signature generation - The application (Lev1) signature, class level (Lev2) signature, and method level (Lev3) signatures [27]

DroidAnalytics is an example of a signature-based detection system [27]. Three different signatures are generated by the detection system, and then compared to known signatures. The system first generates the *Lev3* signature (method signature), which is a signature for each method in the source code. A *Lev2 signature* (class signature or dynamic payload signature) is then generated by the system based upon the signatures generated previously. The *Lev1 signature* (application signature) is generated by the system from the *Lev2* signature. The three levels of signatures are used by DroidAnalytics to generate a similarity score. DroidAnalytics can detect repackaged and mutated malware by comparing the similarity score of the application to known malwares from the database. Figure 5 displays the framework used by DroidAnalytics to generate signatures.

Wang et al. [28] proposed a permission-based detection method for Android applications. The system ranks individual Android permissions according to the security threat level that they pose. Applications that have been analyzed and found to have several risky permissions will be reported as potentially malicious.

B. Dynamic Approach

The dynamic approach to malware detection aims to analyze applications during execution by examining system calls at runtime to differentiate malicious and benign behaviors. Dynamic analysis commonly occurs in a sandbox environment or a virtual machine to avoid accidental infection of physical devices. Dynamic approaches are computationally heavy, and often difficult to implement at the device level due to limited computational power of the device. Using dynamic analysis, common stealth techniques such as code obfuscation are ineffective at preventing detection. Listed below are the advantages and disadvantages of dynamic analysis for malware detection:

Advantages:

- More thorough analysis of application functionality allows detection of unknown malwares
- Dynamic analysis detects stealth techniques such as code obfuscation and dynamic loading
- Some dynamic analysis techniques make real-time malware protection possible
- Does not require access to source code to perform analysis

Disadvantages:

- Computational requirements of dynamic analysis make most solutions infeasible for device level security
- Anti-sandbox and emulation detection evasion techniques have become more common in malware
- Potential for false positives due to similarities between malicious and benign applications

DroidTrace [29] is a ptrace based dynamic analysis system that explores the behavior of dynamic payloads. DroidTrace uses ptrace to monitor system calls of the target process which is running the dynamic payloads. Users can use DroidTrace on their device, avoiding the need for emulation.

RiskRanker [30] automatically assesses the risks from untrusted applications for zero-day malware detection. A two-order risk analysis is the basis for the assessment. The first-order risk assessment determines the presence of high and medium risk applications. Then the second-order risk assessment further investigates suspicious behavior of the high and medium risk applications.

TaintDroid [31] tracks the flow of sensitive data through third-party applications. TaintDroid pressumes by default that downloaded applications are untrusted and monitors in real-time the access and use of users' personal data by the application. Utilization of Android's virtualized architecture enables TaintDroid to have a minimal performance overhead, making it practical for real-time analysis on Android devices.

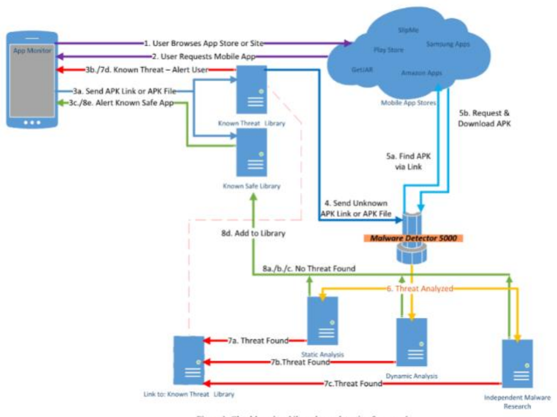

App Monitor
1. User Browses App Store or Site
2. User Requests Mobile App
3b./7d. Known Threat – Alert User
3a. Send APK Link or APK File
3c./8e. Alert Known Safe App
SlipMe
Play Store
Samsung Apps
GetJAR
Amazon Apps
Mobile App Stores
Known Threat Library
Known Safe Library
5b. Request & Download APK
5a. Find APK via Link
4. Send Unknown APK Link or APK File
Malware Detector 5000
8d. Add to Library
8a./b./c. No Threat Found
6. Threat Analyzed
7a. Threat Found
Static Analysis
7b.Threat Found
Dynamic Analysis
7c.Threat Found
Link to: Known Threat Library
Independent Malware Research

Figure 1. Cloud-based mobile malware detection framework

Figure 6: Cloud-Based Architecture [33]

DroidScope [32] is an analysis platform for Android malware analysis. The system builds on top of an emulator and can reconstruct the OS-level and Java-level semantic views. Analysts can then implement custom analysis plugins to aid in the detection of malware. DroidKungFu and DroidDream, two known Android malware samples, were used by the researcherto test the functionality of the system.

C. Cloud Based Detection

The advancements in cloud computing have enabled fast, efficient, and effective mobile security solutions. Utilization of cloud technology enables the computational requirements of malware detection to be in the cloud. Owing to the increase in available computational power, cloud-based solutions can employ multiple static and dynamic detection methods. Listed below are the advantages and disadvantages of cloud-based mobile malware detection:

Advantages:

- Removes computational requirements from the physical device
- Avoids scanning applications client side
- Enables use of multiple detection methods

Disadvantages:

- Cloud provider downtime causes gaps in the availability of the service
- Poor local network connection can limit the performance of scanning capabilities
- Limitations of specific detection methods still apply

Penning et al. [33] proposed a framework for a cloud-based detection system. The detection system has seven main components. The *app monitor* monitors incoming mobile applications and updates. When a new request is detected, the *Malware Detector 5000* (located on the cloud) is notified by the monitor and performs an analysis of the application. The malware detector has five components: the *known threat library*, containing all known malicious applications, the *known safe library*, containing all known safe applications, and three analysis engines. The *dynamic analysis* component performs dynamic analysis of the application, the *static analysis* component performs static analysis of the application, and the *independent research* component analyzes unknown malwares that will reveal details that the static and dynamic engines could not detect on their own. Figure 6 shows the proposed framework of this cloud-based solution.

ScanMe [34] is a cloud-based malware analysis service that performs real-time malware detection. Google Cloud Messaging (GCM) service is used by the application to collect data from mobile devices. The data is then used by the application for threat monitoring and detection. ScanMe uses various static and dynamic analysis methods, as well as offering a pre-configured sandbox environment to scan applications before users install them to their device.

D. Hybrid Based Detection

Hybrid detection systems use the combined force of static and dynamic analysis to greatly improve malware detection capabilities. Hybrid systems often use signature matching for static analysis and anomaly-based detection for dynamic analysis. Listed below are the advantages and disadvantages of hybrid-based detection:

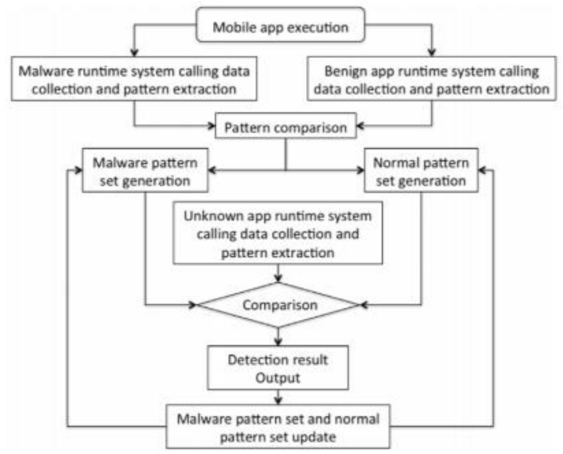

Mobile app execution
Malware runtime system calling data collection and pattern extraction
Benign app runtime system calling data collection and pattern extraction
Pattern comparison
Malware pattern set generation
Normal pattern set generation
Unknown app runtime system calling data collection and pattern extraction
Comparison
Detection result Output
Malware pattern set and normal pattern set update

Figure 7: Detection Procedure [37]

Advantages:

- Hybrid analysis is more effective than static or dynamic approaches alone
- Higher detection accuracy due to combination of approaches
- Can detect new malwares in the wild

Disadvantages:

- New malware variants must be manually added to existing databases by the host
- Most dynamic analysis still requires computational requirements beyond device capabilities

AAsandbox [35] is an example of a hybrid detection system that utilizes sandboxing to perform both static and dynamic analysis. In the sandbox environment, application installation files and corresponding executables are disassembled by the application. AAsandbox then analyzes the files for malicious code fragments and characteristics of previously known malwares. If necessary, the sandbox can then perform dynamic analysis by using an Android emulator, commonly used for testing, and debugging Android applications, to detect potential malicious activities. The use of the emulator requires significant computational resources.

Wang et al. [36] proposed a hybrid mobile malware detection system that makes use of a misuse detector and anomaly detector. The misuse detector is a signature-based detection method maintains a database of all known malicious and benign applications. The anomaly detector is used by the system to perform zero-day malware detection through dynamic analysis. Researchers designed the system for use by application stores and users.

Tong and Yan [37] provided another such hybrid approach to malware detection in Android devices. The hybrid system uses execution data of a set known sample of malware and benign applications to generate patterns of individual system calls. The system can then compare the generated patterns to known applications currently in the database. If the pattern generated does not match a known application in the database, the system can then use a dynamic method to extract system calls to compare against known malicious and benign applications to judge whether the application is malicious or benign. Figure 7 shows the detection procedure used by the system.

E. Machine Learning Approach

Machine learning has been used by antimalware applications for many years, however recent developments have enabled the development of more sophisticated learning models. Researchers can identify malware in the wild faster and more efficiently by using machine learning technology. Machine learning models can classify malware into different categories (trojans, ransomware, adware, etc.) as well as clustering similar malwares to detect behavior in unknown variants. Listed below are the advantages and disadvantages of machine learning enhanced malware detection methods:

Advantages:

- Machine learning enhanced detection methods can quickly identify new malware variants
- Learning models can be continuously improved by researchers/developers
- Learning models are strong against code obfuscation and polymorphic malware
- Learning models can quickly recognize behavioral patterns that would otherwise require significant time investment

Disadvantages:

- Learning models need a large, well-labeled dataset to learn patterns effectively
- Requires human intervention in the case of learning model corruption

Shabtai et al. [38] proposed a malware detection system that uses machine learning to enhance detection of abnormal network behavior in mobile applications. The system monitors applications running on a device, and then utilizes machine learning to train models on "normal" network behavior of the application. Continuous monitoring of the network behavior would then enable the system to detect any behavioral changes in the application's network behavior.

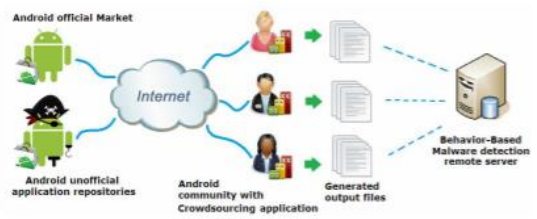

Android official Market
Internet
Android unofficial application repositories
Android community with Crowdsourcing application
Generated output files
Behavior-Based Malware detection remote server

Figure 8: Behavior-Based Malware Detection Framework [41]

CrowDroid [39] is a machine learning enhanced behavior-based malware detection system. CrowDroid offers a lightweight client that monitors Linux Kernel system calls and then sends them to a remote server for processing. A remote server then parses the data and creates a dataset of behavior data. The datasets are then clustered by the application using a partitional clustering algorithm that can differentiate between benign and malicious applications. The system depends on user involvement by submitting behavior-related data of each application they use to the lightweight client. The framework for the behavior-based malware detection is shown in Figure 8.

MaMaDROID [40] detects Android malware by building Markov chains of behavioral models. The system characterizes the sequence of API calls performed by Android apps. The transitions are then modeled by the application as Markov chains. Features can then be extracted, enabling the machine learning algorithms to classify the application as malicious or benign.

DroidSieve [41] uses machine learning to classify obfuscated Android malware quickly and accurately. DroidSieve relies on API calls, code structure, and permissions that are characteristics of Android malware. Extracted features are identified by the application as belonging to one of two classes: resource-centric, which the system obtains from the resources used by the application, and syntactic features obtained from the source code and metadata of the application. The features extracted are common even among obfuscated malwares and can be used by the system to identify malware variants.

F. Deep Learning Approach

Deep learning is a subset of machine learning that aims to simulate animal (human) intelligence. The primary difference between machine and deep learning is that deep learning does not necessarily rely on structured/labeled data to for classification and can automatically extract features and classify data into various classes.

Deep learning is an emerging focus of mobile malware detection researchers. Like machine learning, deep learning is being used by researchers and developers to enhance existing detection methods. Deep learning has enabled the improvements in prediction performance and effectiveness of traditional mobile malware detection methods. The three recent developments in computer technology that have allowed for the use of deep learning based models are: (1) increasing number of labeled malware datasets to security researchers, (2) computational power has become cheaper, (3) researchers have created more tools to create detection models that have resulted in better performance [42]. Listed below are the advantages and disadvantages of deep learning enhanced malware detection methods:

Advantages:

- Provides improved performance with large datasets
- No need for feature engineering

Disadvantages:

- Dependent on the quality of data
- Worse performance on small data sets
- Model training time and cost increased when compared to machine learning models

McLaughlin et al. [42] proposed an Android malware detection system that uses a deep convolutional neural network (CNN). Android applications are disassembled by the system. The raw opcode sequence is then analyzed by the system to perform static analysis. The CNN learns malware features automatically from the disassembled raw opcode sequences. Once the CNN has been trained, the system can be executed using a devices GPU, commonly included in most new Android smartphones.

Deep4MalDroid [43] is a deep learning framework for Android malware detection. The proposed framework uses a novel dynamic analysis method named *Component Traversal* that can execute the code routines of Android applications. From that the extracted Linux kernel system calls can be used by Deep4MalDroid to create a weighted directed graph. The weighted directed graph is then used by the deep learning framework to detect unknown malwares. Commercial Android antimalware applications have implemented the Deep4MalDroid system. Figure 9 shows an example of a neural network with associated weights.

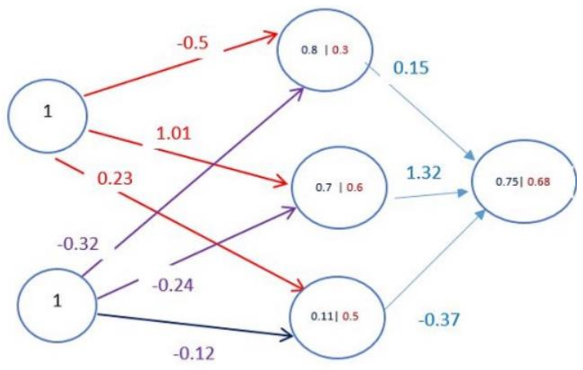


Figure 9: Example of Neural Network with Associated Weights [44]

DDefender [44] is an Android malware detection system using static and dynamic detection methods enhanced by deep learning models. The system consists of two main components: a lightweight application on the user's device that performs dynamic analysis, and a server-side system that performs static analysis. A deep neural network is used by DDefender for feature extraction to achieve a high level of detection accuracy.

G. Blockchain Approach

Blockchain technology originally was used by researchers and developers for securing cryptocurrency transactions, however current research is using blockchain technology in a variety of cybersecurity related fields. Mobile malware detection is one such field. Blockchain technology is being used by researchers and developers to enhance detection methods on both physical devices and at the application store level. Blockchain based approaches utilize various blockchain styles (private, public, consortium) to perform application analysis. Listed below are the advantages and disadvantages of blockchain based mobile malware detection:

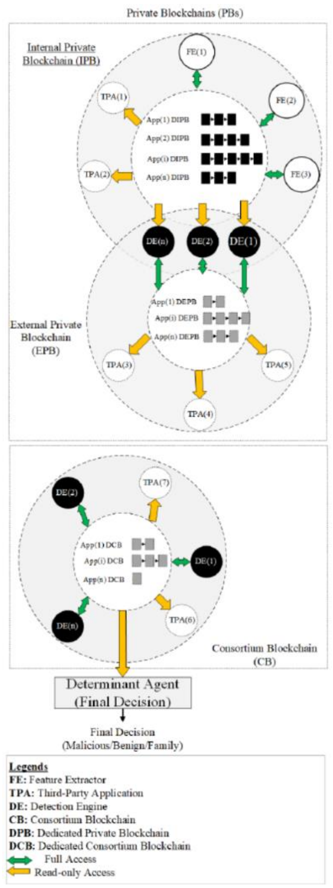
Advantages:

- Enables collaboration of application stores and device users to detect malware more effectively
- Enables the use of a variety of different detection systems
- Decentralized library of known malicious and benign applications

Disadvantages:

- May be difficult to convince competing application stores to collaborate
- Unproved technology as real-world implementations are limited

Homayoun et al. [45] proposed a framework for detecting malicious applications in online mobile app stores using blockchain technology. The framework is composed of a private blockchain and a consortium blockchain. The private blockchain is used by the framework for feature extraction to determine if an application is malicious based on the available feature blocks. Each application in each store would have its own private blockchain, where the specific application store would use their own detection engines to determine the application to be malicious or benign. The application store then adds their application block to the consortium blockchain (where the other application stores are members). If two-thirds of the participating members sign the block, then the application is determined to be malicious or benign based on the information provided in the block. The framework of the system is displayed in Figure 10.



Private Blockchains (PBs)
Internal Private Blockchain (IPB)
FE(1)
FE(2)
FE(3)
TPA(1)
TPA(2)
App(1) DIPB
App(2) DIPB
App(i) DIPB
App(n) DIPB
DE(m)
DE(2)
DE(1)
External Private Blockchain (EPB)
App(1) DEPB
App(i) DEPB
App(n) DEPB
TPA(3)
TPA(4)
TPA(5)
DE(2)
TPA(7)
App(1) DCB
App(i) DCB
App(n) DCB
DE(1)
DE(n)
TPA(6)
Consortium Blockchain (CB)
Determinant Agent (Final Decision)
Final Decision (Malicious/Benign/Family)
Legends
FE: Feature Extractor
TPA: Third-Party Application
DE: Detection Engine
CB: Consortium Blockchain
DPB: Dedicated Private Blockchain
DCB: Dedicated Consortium Blockchain
Full Access
Read-only Access

Figure 10: Private and Consortium Blockchain Framework [45]

CB-MDEE [46] is a framework for mobile malware detection using blockchain technology. Researchers designed the system using *private* and *consortium* blockchains and four distinct layers. The *network layer* consists of linked nodes that communicate through a P2P network. The nodes are servers located in different geographic areas, and the nodes may freely join or exit the network. Nodes in the network continuously exchange block information with neighboring nodes. The *storage layer* stores features of malicious code. Features are stored in blocks that contain information specific to malicious code features and cannot be modified by members of the consortium block chain once signed. The *support layer* provides an interface between users and the malicious code examples in the storage layer. Digital signatures are utilized by the support layer to ensure the legitimacy of signed nodes on the consortium block chain. The *application layer* allows the user to perform malware detection along with digital storage of applications.

Table I. Analysis of Malware Detection Systems

Approach	Name	Year	Deployment	Method	Advantages	Limitations
Static Analysis	*AndroSimilar* [24]	2013	Off-Device	• Generates signature from statistically significant features • Compares signatures to signatures in database • Determines application to be malicious or benign by comparing similarity score	• Able to detect repacked and code obfuscated malware • Minimal false positives when scanning know benign applications	• Database has limited sample of malware signatures • Cannot detect zero-day malware types
	DroidAnalytics [25]	2013	Off-Device	• Uses a three-level signature generation scheme • Compares similarity score to known malicious applications in the database	• Automates the process of malware collection, analysis, and management • Can detect zero-day repackaged malware	• Cannot detect unknown malware types • Similarity score is based on level-2 signature and may classify benign applications as malicious
Dynamic Analysis	*DroidTrace* [27]	2014	On-Device	• First, uses static analysis to discover functions with dynamic loading behavior • Generates an *application control flow graph* to explore possible execution paths • Automatically runs the application via each execution path and then uses ptrace to analyze dynamic payloads	• Forward execution methodology called "physical modification" significantly enhances security analysis process to discover malicious behaviors • Able to detect zero-day payloads	• Only focuses on dynamic loading malwares
	RiskRanker [28]	2012	Off-Device	• Analyzes apps to detect high and medium risk behaviors • First-order modules sifts through non-obfuscated apps to detect risks • Second-order module collects are correlates various signs or patterns of behavior common only to malicious apps	• Can detect zero-day malware • Scalable system to scan entire application store libraries • Detected 322 zero-day malwares from 11 distinct families	• Scalability becomes limited if more complex systems are implemented • Possible for malware developers to counter some of the detection techniques utilized • Dynamic Dalvik code execution scheme has limitations

Approach	Name	Year	Deployment	Method	Advantages	Limitations
	TaintDroid [24]	2015	Off-Device, potential for distributed deployment	• Sensitive device data is automatically tainted to track when it leaves the system • When the data leaves the system, the label of data the application, and the destination are recorded	• Designed for minimal performance overhead • Can simultaneously track multiple sources of sensitive data	• Only tracks explicit dataflows • Trades precision for performance • Significant false positives when the tracked information contains configuration identifiers
	DroidScope [25]	2012	Off-Device	• Analysis is performed outside of the device on an emulator • Four tools are used to perform virtual machine based dynamic analysis	• Analysts can reveal behavior of a malware sample's Java and native components • Analysts can implement custom plugins	• Limited code coverage • Systematic approach to discovering different execution paths would be more desirable • Intrinsic differences between the emulated environment and mobile systems
Cloud-Based Analysis	*Penning et al.* [27]	2014	Off-device	• Monitor detects request to download app and forwards request to malware detector • Checks application with applications in known threat/safe libraries • If unknown, application is passed through detection processors for further analysis	• Places the work outside of the device offloading computational requirements to the cloud • Prevents the mobile device from scanning applications locally • Processors can use more powerful and efficient detection engines	• User is dependent on service provider to maintain uptime of the service • Poor local network connection negatively impacts performance of the scan • Proposed framework rather than actual implemented service
	ScanMe [28]	2016	Distributed	• Data collected from Google Cloud Messaging is delivered to a remote server to perform threat monitoring and detection • User is alerted in malicious activity is detected • Users can also use sandbox environment to test applications	• Developed service requires minimal resources to run on a mobile device • Efficient; takes less than one minute to perform a complete static and dynamic analysis • Offers a pre-configured sandbox environment for users to test and analyze applications before installing	• Limited testing of the system was performed (100 benign and 100 malicious applications) • Original detection sandbox only handles APKs from a single Android client • Collected data needs to be processed and transmitted over mobile networks, which have limited wireless network and computing resources

Approach	Name	Year	Deployment	Method	Advantages	Limitations
Hybrid Analysis	*AAsandbox* [24]	2010	Off-Device	• The code is disassembled to perform static analysis • Investigated applications are installed to emulated environment • Applications are then executed to perform behavioral analysis	• Can be used by app stores to analyze every app that is submitted	• Requires cloud implementation as the system requires significant computational resources • Cannot detect new malware types
	Wang et al. [25]	2015	Off-Device	• Features from the source code are extracted to perform static analysis • If static analysis results in an unknown application, dynamic anomaly detection is used for further investigation	• Designed to be users by both app stores and mobile users • Can detect zero-day and new variants • Generates a detailed analysis report	• Must be located on the cloud due to significant computational requirements
	Tong and Yan. [26]	2017	Off-Device	• Applies both malicious and normal patters to detect malicious applications • Unknown apps are detected by collecting and extracting the dynamic system calls	• Approach is efficient and accurate for malware detection • The data process is based on simple algorithms with low computational costs	• Needs to continuously collect new malware and benign apps to maintain detection accuracy • Not feasible to perform large scale data processing on a mobile device
Machine Learning Enhanced Solutions	*Shabtai et al.* [27]	2014	Distributed	• Client-side software extracts features related to network behavior of installed apps • Client-side software can then detect deviations from normal behavior and alert the user via the GUI • Server-side client is used to do a robust anomaly detection of applications before download and installation	• Low false positive rate • Relatively low overhead on the mobile device's system resources • Fine-tuning of feature evaluation could further improve accuracy	• Detection is based only on applications network patterns • Feature extractor and the aggregation processes impact on the device's resources were not evaluated
	CrowDroid [28]	2011	Distributed	• Client app performs system calls tracing • Log file is created and sent to remote server for dynamic analysis	• Allows for real-time detection via client app • High detection accuracy	• Relies on assumption that malicious apps invoke more system calls

Approach	Name	Year	Deployment	Method	Advantages	Limitations
	MaMaDroid [24]	2017	Off-Device	• Markov chains are used to model the behavior of apps through a sequence of API calls • Features are extracted from the Markov chains and the machine learning model can then classify apps as malicious or benign	• System is scalable as every single app is modeled independently from others • App features can easily be appended in a new training set • Significantly higher accuracy with reasonable run times compared to similar solutions	• Requires significant amount of memory to perform classification due to large number of features • Cannot detect new malware types
	DroidSieve [25]	2017	Off-Device	• Relies on API calls, code structure, permissions, and the set of invoked components to perform static analysis • Importance of features was analyzed to determine the best features to use for detection	• Deployable for obfuscation-resilient detection at scale • Time required to process the test samples was negligible	• Detection technique may not be robust against mimicry attacks as the features that are used can be contrived • Performance of the classifiers might degrade over time as malware becomes more sophisticated
Deep Learning Enhanced Solutions	*McLaughlin et al.* [27]	2017	Distributed	• Disassembled bytecode is of an application is treated as text to be analyzed • Features are learned automatically from raw data	• Computationally efficient as training and testing time in linearly proportional to the number of malware samples • Can scan large numbers of malware files per second using smartphone GPUs' • Can be easily retrained with new malware samples	• Focuses only on static analysis in current state • Further research into appropriate data augmentation schemes for malware detection is necessary
	Deep4MalDroid [28]	2016	Distributed	• *Component Traversal* is used to automatically execute code routines of each Android app • Weight directed graphs are constructed from results • Deep learning framework is used for model construction	• Able to detect unknown malware types • Implemented into existing Android anti-malware software • Proposed method outperforms existing solutions	• Only designed for dynamic analysis

Approach	Name	Year	Deployment	Method	Advantages	Limitations
	DDefender [24]	2018	Distributed	• A lightweight application running on the user's device is used to perform dynamic analysis • Server side, a system is in place to perform static analysis • DDfender automates the entire process and blocks malicious applications in real time	• Over 1000 features used to classify applications • Analysis is done on the user's device	• The dataset was a 50/50 split of malicious and benign apps which is not indicative of real-world parameters
Blockchain Based Solutions	*Hamayoun et al.* [27]	2014	Off-Device	• Feature extraction is integrated into a private blockchain to extract features that can be fed into a detection engine to determine if an app is malicious or benign • The consortium block chain provides a mechanism of final decision making by considering the results of all detection engines	• Enables the collaboration between different app stores • Multiple detection engines are used to determine the malicious nature of an app • Implementation is not limited to a specific machine learning algorithm	• May be difficult to convince competing app stores to collaborate • Unproven technology as it is only a proposed framework
	CB-MDEE [28]	2018	Distributed	• The consortium blockchain is composed of members in distributed detection organizations • The public blockchain is the application chain, open to for any user who needs to provide detection and evidence services for joining as a member node	• Enables app stores to collaborate with mobile device users to provide more data for malware detection • Can be used to detect unknown malwares	• Requires users to install a customized application for data collection

V. Discussion

The significant market share possessed by Android operating systems means that malware developers will continue to target Android in search of financial gains. It is necessary for Android malware detection techniques to improve as malware developers also continuously improve their evasion techniques. The increasingly large amount of malware samples makes has made it difficult for traditional signature-based solutions to maintain pace. Dynamic approaches continue to be effective, but with Android user's reluctance to dedicate the necessary computational space on their personal device, dynamic solutions are not able to effectively counter malware developers. Table 1 provides an analysis of existing and emerging Android malware detection systems.

The emergence of machine learning, deep learning, and blockchain based solutions have enabled malware detection to become faster, more efficient, and more accurate. However, these technologies are also limited by the computational constraints associated with mobile devices. Utilization of cloud technology has made it possible to implement computationally heavy solutions in a manner that is feasible for mobile device users to implement as a solution for detecting malicious applications. Cloud-based solutions are still limited in the fact that they remain reliant on third-party vendors to maintain service uptime.

The most effective malware detection solution will be one that takes advantage of the computational power available in the cloud to implement advanced solutions, as well as provide a robust solution that is available on the device as a failsafe in an event where the cloud service in not available. This solution would be a multilayered defense that allows mobile device users to be certain that their devices are always protected from malicious activities.

Application stores should also refine and improve their detection systems. The use of blockchain technology for a decentralized collaborative solution is promising. Blockchain solutions enable a collaborative effort for malware detection using the consortium blockchain. The design of the consortium blockchain makes it inherently resistant to tampering because it requires a set number of participants to sign and approve a block before it is added to the blockchain. This prevents a scenario where a malware developer can infiltrate the consortium blockchain and convince the members to approve their malicious application.

VI. Conclusion

Malware threats will continue to pose a significant threat to Android users because of the fragmented nature of the manufacturing ecosystem and the opportunity for financial gain. Section II discusses the most prevalent malware threats to Android devices currently. Section III explored the various malware penetration methods, including evasion techniques for avoiding detection, used by malware developers. Section V provided a survey of existing and emerging malware detection systems. Section V proposed a multilayered malware detection solution that would be best suited for a comprehensive security system on Android devices.

References

[1] E. Protalinski, “Android Passes 2.5 Billion Monthly Active Devices,” May 7, 2019. [Online]. Available: https://venturebeat.com/2019/05/07/android-passes-2-5-billion-monthly-active-devices/

[2] M. Chau and R. Reith, “Smartphone Market Share,” April 2, 2020. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[3] S. Rexaline, “10 Years of Android: How The Operating System Reached 86% Market Share,” September 25, 2018. [Online]. Available: https://www.benzinga.com/news/18/09/1 2404475/10-years-of-android-how-the- operating-system-reached-86-market-share

[4] L. Tung, “Android Fragmentation: There are now 24,000 Devices from 1,300 Brands,” August 6, 2015. [Online]. Available: https://www.zdnet.com/article/android-fragmentation-there-are-now-24000-devices-from-1300-brands/

[5] J. Clement, “Number of Available Application in the Google Play Store from December 2009 to March 2020,” March 19, 2020. [Online]. Available: https://www.statista.com/statistics/26621 0/number-of-available-applications-in-the-google-play-store/

[6] A. Sharma, “Top Google Play Store Statistics 2019-2020 You Must Know,” July 22, 2019. [Online]. Available: https://appinventiv.com/blog/google-play-store-statistics/

[7] R. Lemos, “Google Bounder Vulnerabilities Probed by Security Researchers,” n.d. [Online]. Available: https://www.eweek.com/security/google- bouncer-vulnerabilities-probed-by-security-researchers

[8] W. Zhou et al., “Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces,” in *Proceedings of the Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, Texas,* February 7-9, 2012. [Online]. Available: https://doi.org/10.1145/2133601.2133640

[9] “Malicious Android App Had More Than 100 Million Downloads in Google Play,” August 27, 2019. [Online]. Available: https://www.kaspersky.com/blog/camscanner-malicious-android-app/28156/

[10] K. Beckert-Plewka et al., “Mobile Malware Report – No Let-Up with Android Malware,” July 30, 2019. [Online]. Available: https://www.gdatasoftware.com/news/2019/07/35

228-mobile-malware-report-no-let-up-with-android-malware

[11] "Android Banking Trojans: History, Types, Modus Operandi," January 14, 2020. [Online]. Available: https://www.tripwire.com/state-of-security/security-data-protection/android-banking-trojans-history-types-modus-operandi/#:~:text=The%20first%20one%20was%20the,to%20bypass%20two%2Dfactor%20authentication.

[12] "Trojan-SMS.ANDROIDOS.FAKEINST," 2015. [Online]. Available: https://threats.kaspersky.com/en/threat/Trojan-SMS.AndroidOS.FakeInst/

[13] J. Shilko. "New Variant of BankBot Banking Trojan Ups Ante, Cashes Out on Android Users," March 13, 2018. [Online]. Available: https://info.phishlabs.com/blog/new-variant-bankbot-banking-trojan-aubis

[14] R. Lipovský et al., "The Rise of Android Ransomware," 2016. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2016/02/Rise_of_Android_Ransomware.pdf

[15] "Koler 'Police' Mobile Ransomware," 2014. [Online]. Available: https://usa.kaspersky.com/resource-center/threats/koler-police-ransomware-virus

[16] A. Brandt, "Automated Android Attacks Deliver "UFO" Cryptominer Trojan," February 26, 2019. [Online]. Available: https://news.sophos.com/en-us/2019/02/26/automated-android-attacks-deliver-ufo-cryptominer-trojan/

[17] R. Shields, "REPORT: Ad Fraud is 'Second Only to the Drugs Trade' as a Source of Income for Organized Crime," June 6, 2016. [Online]. Available: https://www.businessinsider.com/wfa- report-ad-fraud-will-cost-advertisers-50-billion-by-2025-2016-6

[18] E. Root and A. Polkovnichenko, "SimBad: A Rogue Adware Campaign on Google Play," March 13, 2019. [Online]. Available: https://research.checkpoint.com/2019/simbad-a-rogue-adware-campaign-on-google-play/

[19] "Trojan:Android/DroidDream.A," n.d. [Online]. Available: https://www.f-secure.com/v-descs/trojan_android_droiddream_a.shtml

[20] W. Hijawi et al., "Toward a Detection Framework for Android Botnet," in *International Conference on New Trends in Computing Sciences, ICTCS 2017,* September 2017. [Online]. Available: 10.1109/ICTCS.2017.48

[21] "Android Repackaging Attack Lab," 2015. [Online]. Available: http://www.cis.syr.edu/~wedu/seed/Labs_Android5.1/Android_Repackaging/Android_Repackaging.pdf

[22] P. Yan, "A Look at Repackaged Apps and their Effect on the Mobile Threat Landscape," July 15, 2014. [Online]. Available: https://blog.trendmicro.com/trendlabs-security-intelligence/a-look-into-repackaged-apps-and-its-role-in-the-mobile-threat-landscape/

[23] "Trojan:Android.DroidKungFu.C," n.d. [Online]. Available: https://www.f-secure.com/v-descs/trojan_android_droidkungfu_c.shtml

[24] X. Jiang, "Security Alert: New DroidKungFu Variant – Again! – Found in Alternative Android Markets," May 17, 2011. [Online]. Available: https://www.csc2.ncsu.edu/faculty/xjiang4/DroidKungFu3/

[25] "Drive-By-Download," n.d. [Online]Available: https://www.trendmicro.com/vinfo/us/security/definition/drive-by-download

[26] Faruki et al., "AndroSimilar: Robust Statistical Feature Signature for Android Malware Dection," in *Proceedings of the 6th International Conference on Security Information and Networks, SIN 2013, Aksaray, Turkey,* November 26-28, 2013. [Online]. Available: https://doi.org/10.1145/2523514.2523539

[27] M. Zheng et al., "Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware," in *International Conference on Trust, Security and Privacy in Computing and Communications, IEEE 2013, Melbourne, Australia,* July 16-18 2013. [Online]. Available: 10.1109/TrustCom.2013.25

[28] M. Zheng et al., "DroidTrace: A ptrace based Android Dynamic Analysis System with Forward Execution Capability," in *International Wireless Communications and Mobile Computing Conference, IWCMC 2014, Nicosia, Cyprus,* August 4-8, 2014. [Online]. Available: 10.1109/IWCMC.2014.6906344

[29] M. Grace et al., "RiskRanker: Scalable and Accurazte Zero-day Android Malware Detection," in *Proceedings of the 10th International Conference on Mobile Systems, MobiSys 2012, Low Wood Bay, Lake District, UK,* June 25-29, 2012. [Online]. Available: https://doi.org/10.1145/2307636.2307663

[30] W. Enck et al., "TaintDroid: An Information-Flow Tracking System for Realtime privacy Monitoring on Smartphones," *ACM Transactions on Computer Systems,* vol. 32, no. 2, June 2015. [Online]. Available: https://doi.org/10.1145/2619091

[31] L. Yan and H. Yin, “DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis,” in *21st USENIX Security Symposium, USENIX 2012, Washington, USA,* August 8-10, 2012. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/yan

[32] N. Penning et al., “Mobile Malware Security Challenges and Cloud-Based Detection,” in *International Conference on Collaboration Technologies and Systems, CTS 2014, Minnesota, USA,* May 19-23, 2014. [Online]. Available: 10.1109/CTS.2014.6867562

[33] H. Zhang et al., “ScanMe Mobile: A Cloud-Based Android Malware Analysis Service,” *ACM SIGAPP Applied Computing Review,* vol. 16, no. 1, April 2016. [Online]. Available: https://doi.org/10.1145/2924715.2924719

[34] T. Bläsing et al., “An Android Application Sandbox System for Suspicious Software Detection,” in *5th International Conference on Malicious and Unwanted Software, Lorraine, France,* October 19-20, 2010. [Online]. Available: 10.1109/MALWARE.2010.5665792

[35] X. Wang et al., “A Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection with Misuse Detection,” in *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services, MCS 2015, Paris, France,* September 7-11, 2015. [Online]. Available: https://doi.org/10.1145/2802130.2802132

[36] F. Tong and Z. Yan, “A Hybrid Approach of Mobile Malware Detection in Android,” *Journal of Parallel and Distributed Computing,* vol. 103, May 2017. [Online]. Available: https://doi.org/10.1016/j.jpdc.2016.10.012

[37] A. Shabtai et al., “Mobile Malware Detection Through Analysis of Deviations in Application Network Behavior,” *Computers & Security,* vol. 43, June 2014. [Online]. Available: https://doi.org/10.1016/j.cose.2014.02.009

[38] I. Burguera et al., “Crowdroid: Behavior-Based Malware Detection System for Android,” in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 2011*, October 17, 2011. [Online]. Available: https://doi.org/10.1145/2046614.2046619

[39] E. Mariconti et al., “MaMaDroidL Detecting Android Malware by Building Markov Chains of Behavioral Models,” in *Proceedings of 24th Network and Distributed System Security Symposium, NDSS 2017, California, USA,* November 20, 2017. [Online]. Available: https://arxiv.org/pdf/1612.04433.pdf

[40] G. Suarez-Tangil et al., “DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware,” in *Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy, CODASPY 2017, Arizona, USA,* March 22-24, 2017. [Online]. Available: https://doi.org/10.1145/3029806.3029825

[41] N. McLaughlin et al., “Deep Android Malware Detection,” in *Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy, CODASPY 2017, Arizona, USA,* March 22-24, 2017. [Online]. Available: https://doi.org/10.1145/3029806.3029823

[42] S. Hou et al., “Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux System Call Graphs,” in *IEEE/WIC/ACM International Conference on Web Intelligence WorkShops, WIW 2016, Nebraska, USA,* October 13-16, 2016. [Online]. Available: 10.1109/WIW.2016.040

[43] M. Hassankashi, “A Simple and Complete Explanation of Neural Networks,” April 3, 2019. [Online]. Available: https://www.codeproject.com/Articles/1200392/A-Simple-and-Complete-Explanation-of-Neural-Networ

[44] H. Alshahrani et al., “DDefender: Android Application Threat Detection using Static and Dynamic Analysis,” in *IEEE International Conference on Consumer Electronics, ICCE 2018, Nevada, USA,* January 12-14, 2018. [Online]. Available: 10.1109/ICCE.2018.8326293

[45] S. Homayoun et al., “A Blockchain-Based Framework for Detecting Malicious Mobile Applications in App Stores,” in *IEEE Canadian Conference of Electrical and Computer Engineering, CCECE 2019, Alberta, Canada,* May 5-8, 2019. [Online]. Available: 10.1109/CCECE.2019.8861782

[46] J. Gu et al., “Consortium Blockchain-Based Malware Detection in Mobile Devices,” *IEEE Access*, vol. 6, February 13, 2018. [Online]. Available: 10.1109/ACCESS.2018.2805783